ACQUISITION OF ORDINAL WORDS USING WEAKLY SUPERVISED NMF

Vincent Renkens, Steven Janssens, Bart Ons, Jort F. Gemmeke, Hugo Van hamme

Department of Electrical Engineering-ESAT, KU Leuven, Leuven, Belgium
(vincent.renkens, steven.janssens)@student.kuleuven.be
(bart.ons, jort.gemmeke, hugo.vanhamme)@esat.kuleuven.be

ABSTRACT

This paper issues in the design of a vocal interface for a robot that can learn to understand spoken utterances through demonstration. Weakly supervised non-negative matrix factorization (NMF) is used as a machine learning algorithm where acoustic data are augmented with semantic labels representing the meaning of the command. Many parameters that the robot needs in order to execute the commands have an ordinal structure. Constrained subspace NMF (CSNMF) is proposed as an extension to NMF that aims to better deal with ordinal data and thus increase the learning rate of the grounding information with an ordinal structure. Furthermore automatic relevance determination is used to deal with model order selection. The use of CSNMF yields a significant improvement in the learning rate and accuracy when recognising ordinal parameters.

Index Terms— Language acquisition, Machine learning, Non-negative Matrix Factorization (NMF), Ordinal data, Automatic Relevance Determination (ARD)

1. INTRODUCTION

Spoken human-machine interfaces traditionally rely on a speech recognition component that is equipped with a vocabulary and grammar that are fixed prior to deployment of the system [1]. This is an adequate approach for e.g. directory assistance or travel reservation systems where for instance the list of airports is known in advance. In this work, we are however interested in applications where it is too expensive, too elaborate or simply not possible to decide on a vocabulary and grammar prior to deployment, but instead want to learn these components while the user is operating the system. This is a relevant problem in e.g. service robotics where users may refer to objects and actions through words and phrases that can only be interpreted in their specific environment (e.g. "Get a Jupiler from the storage board in the small room." where "Jupiler" is a product name, and "storage board" and "small room" have a particular meaning in the house of the user only). The required vocabulary is hence difficult to finalize in a development phase of the spoken interface. As an alternative, this research investigates a different approach where vocabulary and grammar are *acquired* from a spoken command augmented with a demonstration. A spoken interface based on language acquisition has the additional advantage that users may speak their dialect (or any language) or can have speech disorders. Furthermore, users can choose their own wordings, i.e. the machine adapts to the user rather than the user learning the vocabulary that the machine understands.

In prior work, we have proposed a method for language acquisition through demonstration [2] and have shown that *categorical* concepts can be successfully acquired. An example is home automation where e.g. one of many doors can be selected to be opened or closed. In this work, we focus on the use case where *ordinal* concepts need to be learned, like in control of a robot. A user might give commands like 'Drive a little bit forward'. However, it is hard to predict the exact wordings, nor what this particular user means with 'a little bit'. Therefore, we assume the user demonstrates what he/she means with the command. The user will probably not demonstrate the command in exactly the same way every time, so there will be variation in the data. If a machine learning algorithm is used that does not take the ordinality of the parameters into account these small variations in the demonstration will be interpreted as different commands. This is not optimal and will decrease the learning speed. However, when the algorithm does incorporate the ordinal structure in the learning scheme it can detect that these variations in the demonstration are close to each other and thus they can be grouped under one single command. This way the robot can map expressions like 'a little bit' to a distribution over distances.

Vocabulary and grammar acquisition by demonstration and continuous speech input requires a different approach than traditional hidden Markov models (HMMs). Indeed, in this setting there will be no verbatim transcription of the user's utterance allowing supervised HMM learning. Instead, there is only a *weak* form of supervision in the form of the demonstration. Therefore, weakly supervised Non-negative Matrix Factorization (NMF) is used here as a classifier algorithm. NMF is a popular unsupervised machine learning algorithm where the data is approximated as a linear combination of non-negative latent components [3]. NMF has shown to give good results when compared to other state-of-the-art methods, like e.g. HMMs [4]. In the training phase the latent components that can best approximate the data are found and later used in the testing phase to analyse unseen data in terms of the latent components. To use NMF as a classification algorithm the data is expanded with class labels making it a (weakly) supervised machine learning algorithm. This algorithm has shown to give good results in a.o. speech recognition [5].

The research in this work is funded by IWT-SBO grant 100049 (ALADIN).

A first contribution of this paper is to propose and evaluate an extension to weakly supervised NMF that takes the ordinal structure of the class label data into account. A second contribution is to augment it with automatic model order selection through a Bayesian formulation inspired by automatic relevance determination proposed in [6]. Automatic model order selection is necessary in the design of the robot because it is not known in advance how many commands will be taught to the robot. The user may just use one command to drive forward or he may make a distinction in how far ('a little bit', 'a meter', 'far ahead', ...) or how fast ('slowly', 'quickly') the robot has to drive. Neither is it known if the user will use different wordings for the same command. The model order must be large enough to be able to classify all these commands but not too large to avoid overfitting.

Classifying data with an ordinal structure has been treated in many fields of machine learning using methods including Gaussian Processes [7], Support Vector machines [8] and Neural networks [9]. Classifying ordinal data has many applications like assessing medical risk [10], classifying the grading done by a teacher [7], or in the field of speech technology, classifying speakers in age groups [11]. Notice however that the data in the current work have a particular structure: users use *different* words to denote concepts with an ordinal structure. These different words lead to acoustic feature vectors that occupy distinct regions in the feature space and hence a regression model is not appropriate. Instead, the proposed method will be derived from weakly supervised NMF which will discussed in more detail in section 2. The problem with ordinal data in NMF is discussed and Constrained Subspace NMF (CSNMF) is proposed as a possible solution for this problem in section 3. In section 4 the experimental setup is described and the results of these experiments are discussed in section 5. Finally some conclusions will be formulated in section 6.

2. WEAKLY SUPERVISED NMF

In Non-negative matrix Factorization (NMF), a high dimensional non-negative $F \times N$ matrix $\boldsymbol{V}$ is decomposed in two non-negative matrices of lower rank: an $F \times K$ matrix $\boldsymbol{W}$ and a $K \times N$ matrix $\boldsymbol{H}$. In our setting, $\boldsymbol{V}$ is called the data matrix and every column is a data point. $\boldsymbol{W}$ is called the vocabulary matrix and its columns are the latent components of the data matrix. The elements of $\boldsymbol{H}$ are the activations of these components. The dimensions of these two matrix factors are much smaller than the dimensions of the data matrix ($FK + KN \ll FN$). Because the dimensions of the matrix factors are lower, the decomposition is in general only an approximation of the data matrix:

$$\boldsymbol{V} \approx \boldsymbol{W}\boldsymbol{H} \tag{1}$$

The matrix factors are found with a multiplicative updating scheme that monotonically decreases a cost function [3]. Two popular choices of costs function are the Euclidean distance and the Kullback-Leibler divergence.

In the case of *categorical* data (e.g. which door to open or close; cfr. supra), supervision information is added by expanding the columns of $\boldsymbol{V}$ with a semantic label indicator vector of the same dimension as the number of categories. The label vector contains a one in the position of the corresponding category and zero otherwise. In this paper weakly supervised NMF will be used to categorize spoken commands given to a robot. The vectors of the data matrix $\boldsymbol{V}$ are acoustic feature vectors ($\boldsymbol{v}_{\mathrm{a}}$) expanded with a semantic label vector ($\boldsymbol{v}_{\mathrm{c}}$) [5]:

$$\boldsymbol{v} = \begin{bmatrix} \boldsymbol{v}_{\mathrm{c}} \\ \boldsymbol{v}_{\mathrm{a}} \end{bmatrix} \tag{2}$$

$\boldsymbol{v}$ is a column from $\boldsymbol{V}$. The semantic information will in general contain multiple categorical concepts corresponding to the command type and the command parameters. For example, the command type specifies it's a command about doors (as opposed to lights), the command parameters specify which door is acted upon and what should happen (open or close). Therefore, $\boldsymbol{v}_{\mathbf{c}}$ is constructed by stacking the semantic label vectors [12] of all command parameters of all command types:

$$\boldsymbol{v}_{\mathrm{c}} = \begin{bmatrix} (\boldsymbol{v}_{\mathrm{c}}^{1})^T & (\boldsymbol{v}_{\mathrm{c}}^{2})^T & \cdots & (\boldsymbol{v}_{\mathrm{c}}^{S})^T \end{bmatrix}^T \tag{3}$$

S is the number of different command types. Every vector corresponding to a command type is then represented as:

$$\boldsymbol{v}_{\mathrm{c}}^{s} = \begin{bmatrix} (\boldsymbol{v}_{\mathrm{c}}^{s,1})^T & (\boldsymbol{v}_{\mathrm{c}}^{s,2})^T & \cdots & (\boldsymbol{v}_{\mathrm{c}}^{s,M_s})^T \end{bmatrix}^T \tag{4}$$

M_s is the number of parameters belonging to the command type s. Every vector $\boldsymbol{v}_{\mathrm{c}}^{s,j}$ contains all zeros except for a 1 in the location corresponding to the value of the command parameter. As an example equations (5) shows $\boldsymbol{v}_{\mathrm{c}}$ when there is one command type, move, which has two parameters, distance and velocity. They each have five possible values and the first value is chosen for both parameters. The corresponding command could be: "slowly drive a little bit forward".

$$\boldsymbol{v}_{\mathrm{c}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{5}$$

The classifier is trained by decomposing a training data matrix $\boldsymbol{V}^{\mathrm{trn}}$ with columns constructed as in (2) into a vocabulary matrix $\boldsymbol{W}$ and a training activation matrix $\boldsymbol{H}^{\mathrm{trn}}$. $\boldsymbol{V}^{\mathrm{trn}}$ and $\boldsymbol{H}^{\mathrm{trn}}$ are then discarded because they are data specific and only the vocabulary matrix $\boldsymbol{W}$ is retained. The vocabulary matrix is split up into an upper and a lower part. The lower part $\boldsymbol{W}_{\mathrm{a}}$ represents the acoustic feature vocabulary matrix and the upper part $\boldsymbol{W}_{\mathrm{c}}$ represents the semantic label vocabulary matrix. To classify an unknown command with acoustic feature vector $\boldsymbol{v}_{\mathrm{a}}^{\mathrm{tst}}$ it is decomposed using the known acoustic feature vocabulary matrix:

$$\boldsymbol{v}_{\mathrm{a}}^{\mathrm{tst}} \approx \boldsymbol{W}_{\mathrm{a}}\boldsymbol{h}^{\mathrm{tst}} \tag{6}$$

The unknown semantic labels corresponding to the command can then be approximated using the acquired activations in $\boldsymbol{h}^{\mathrm{tst}}$ and the semantic label vocabulary matrix:

$$\boldsymbol{v}_{\mathrm{c}}^{\mathrm{tst}} \approx \boldsymbol{a} = \boldsymbol{W}_{\mathrm{c}}\boldsymbol{h}^{\mathrm{tst}} \tag{7}$$

Using $\boldsymbol{a}$, called the activation vector, the command can be classified. First the command type is determined by checking which type has the largest total activation in its corresponding vector ($\boldsymbol{v}_{\mathrm{c}}^{s}$). Next the parameters are determined as the weighed average of the possible values with the activations in their corresponding vectors ($\boldsymbol{v}_{\mathrm{c}}^{s,p}$) as weights.

3. PROPOSED METHOD

3.1. Constrained Subspace NMF

As mentioned in the previous section the matrix factors are found by minimizing some cost function:

$$D(\boldsymbol{V}||\boldsymbol{W}\boldsymbol{H}) = \sum_{f=1}^{F}\sum_{n=1}^{N} D(V_{fn}||(WH)_{fn}) \tag{8}$$

In this expression the problem with using ordinal data in NMF already becomes apparent. If the columns of $\boldsymbol{V}$ are constructed as in (2), the distance between two semantic label vectors calculated in (8) is independent of their relative position in the ordinal structure. It does not matter if the position of the 1 representing the parameter values is close to the position of the 1 in the other vector or not. This means that classifying a command to a class that is far away in the ordinal structure will be penalized equally hard as classifying it to a class that is close by in the ordinal structure. To solve this issue, the NMF problem will be treated as a likelihood maximization problem. Many choices of cost functions correspond to an assumption of the distribution of the noise on the data [13]. The Kullback-Leibler divergence, which is used in this paper, corresponds to a Poisson distribution. To obtain the formula's proposed in [3] the likelihood of the data is written as:

$$P(\boldsymbol{V}|\boldsymbol{W},\boldsymbol{H}) = \prod_{f=1}^{F}\prod_{n=1}^{N} P(V_{fn}|(WH)_{fn}) \tag{9}$$

This means that the likelihood of the data depends on the corresponding element in the approximation and not on its neighbours. If the data is not ordinal this may be a reasonable assumption. However, when working with ordinal data this is suboptimal. This would mean that when a user demonstrates the command 'drive a little bit forward' twice with distances (command parameters) that are not exactly the same, the commands would bare no relation. This way it is hard to learn the meaning of 'a little bit'. The expression of the likelihood should thus be changed to take the ordinal structure of the command parameter values (called *categories*) into account, i.e. that neighbouring categories should be likely substitutions. Information about these ordinal proximity relation is introduced via a non-negative $F \times P$ matrix $\boldsymbol{L}$. $\boldsymbol{L}$ is called the *interdependency matrix* and its elements denote how dependent the elements in the data matrix are on the elements in the approximation. L_{fp} denotes how dependent V_{fn} is on $(WH)_{pn}$. The likelihood of the data matrix is now written as:

$$P(\boldsymbol{V}|\boldsymbol{W},\boldsymbol{H}) = \prod_{f=1}^{F}\prod_{n=1}^{N} P(V_{fn}|\sum_{p=1}^{P} L_{fp}(WH)_{pn}) \tag{10}$$

Maximizing this likelihood corresponds to approximating the data matrix as:

$$\boldsymbol{V} \approx \boldsymbol{L}\boldsymbol{W}\boldsymbol{H} \tag{11}$$

This is a three-factor NMF with one constant matrix. Non-smooth NMF proposed in [14] is also a three-factor NMF but in this method the middle matrix is constant. Filtering-NMF [15] is a method where the third matrix is constant. Though mathematically equivalent to the current formulation through matrix transposition, its interpretation is quite different since filtering NMF smooths over different *observations* (at subsequent times in [15]) while here different *features* are combined. Therefore, the term Constrained Subspace NMF (CSNMF) is coined here. In NMF the vectors in the approximation can lay everywhere in the positive orthand. However, by adding the constant matrix $\boldsymbol{L}$ the vectors of the approximation can only lay in the subset of the positive orthand spanned by the columns of $\boldsymbol{L}$.

3.2. Choice of the interdependency matrix

For the design of the robot a square interdependency matrix is chosen ($P = F$). The elements in the interdependency matrix $\boldsymbol{L}$ denotes how dependent categories are on the other categories in the same ordinal structure. There should be no influence from categories in another ordinal structure so the elements of $\boldsymbol{L}$ corresponding to these interdependencies should be zero. The highest influence should come from categories that are close in the ordinal structure. This means that the highest values in $\boldsymbol{L}$ should be concentrated around the diagonal. The width of this peak determines how much influence the categories close in the ordinal structure have. This is dependent on how the data was generated. In the case of demonstrations for a robot, the user generates the parameter values. If the user is very precise in his or her demonstrations, the peak should be thin because for the same command only categories close to each other will be selected. However, when the user is not very precise there will be a lot of variability in the supervision information and thus many different categories for the same command will be selected. The peak should then be wide because all these categories should influence each other. In general the columns in $\boldsymbol{L}$ should resemble the demonstration behaviour of the user and would require a user study. It will be experimentally shown below that the actual values in $\boldsymbol{L}$ are not critical to the performance of the method. A gamma distribution was chosen to model the behaviour at the boundaries. To get the same kind of distributions for the lower end categories as the higher end categories, the distributions for the higher end categories are mirrored versions of the ones from the lower end categories. The number of values is finite so the distributions are cut off and renormalized to sum to one. As an example several columns of $\boldsymbol{L}$ are plotted for a parameter that can take 201 values in figure 1 (only the non zero values are plotted).

3.3. Automatic relevance determination

Automatic relevance determination (ARD) is used to achieve model order selection. The method proposed in [6] is adopted in this paper. A Bayesian treatment of NMF is done where the relevance of the columns of $\boldsymbol{W}$ and the rows of $\boldsymbol{H}$ are inserted as latent variables $\boldsymbol{\lambda}$. The probability of the latent variables can be written as:

$$P(\boldsymbol{W},\boldsymbol{H},\boldsymbol{\lambda}|\boldsymbol{V}) = \frac{P(\boldsymbol{V}|\boldsymbol{W},\boldsymbol{H})P(\boldsymbol{W}|\boldsymbol{\lambda})P(\boldsymbol{H}|\boldsymbol{\lambda})P(\boldsymbol{\lambda})}{P(\boldsymbol{V})} \tag{12}$$

where $\boldsymbol{\lambda}$ is a vector of length K. λ_k is a variance like parameter that corresponds to the coupled relevance of the k^{th} column of $\boldsymbol{W}$ and the k^{th} row of $\boldsymbol{H}$. The noise on the elements of the data matrix is assumed to be Poisson distributed. This might not be the best assumption for the semantic labels because they can only be 0 or 1. A Binominal distribution seems better suited but this is left for further research. Exponential priors are assumed on both $\boldsymbol{W}$ and $\boldsymbol{H}$.

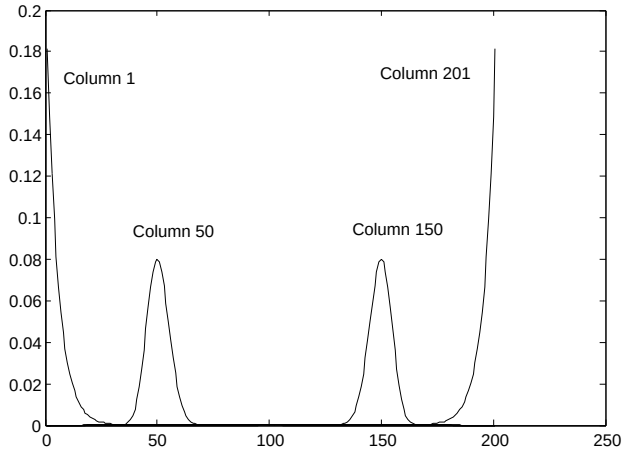

Fig. 1. Some examples of the columns of $\boldsymbol{L}$

An inverse gamma distribution is assumed on $\boldsymbol{\lambda}$. The multiplicative updating scheme that maximizes this probability with the likelihood of the data as in (10) are:

$$W_{pk} = W_{pk} \frac{\sum_{f=1}^{F} \sum_{n=1}^{N} L_{fp} \frac{V_{fn}}{(LWH)_{fn}} H_{kn}}{\sum_{f=1}^{F} \sum_{n=1}^{N} L_{fp} H_{kn} + \frac{1}{\lambda_k}} \tag{13}$$

$$H_{kn} = \frac{\sum_{f=1}^{F} \frac{V_{fn}}{(LWH)_{fn}} (LW)_{fk}}{\sum_{f=1}^{F} (LW)_{fk} + \frac{1}{\lambda_k}} \tag{14}$$

$$\lambda_k = \frac{\sum_{f=1}^{F} (W_{fk}) + \sum_{n=1}^{N} (H_{kn}) + b}{M + N + a + 1} \tag{15}$$

where a and b are the fixed parameters of the inverse gamma distribution of $\boldsymbol{\lambda}$. They are chosen as described in [6], a is chosen small compared to $F + N$ and b is chosen as:

$$b = \sqrt{\frac{(a-1)(a-2)\hat{\mu}_{\boldsymbol{V}}}{K}} \tag{16}$$

where $\hat{\mu}_{\boldsymbol{V}}$ is the global empirical mean of $\boldsymbol{V}$. During the learning process, the columns of $\boldsymbol{W}$ and the rows of $\boldsymbol{H}$ are removed if their relevance is low to speed up the learning process. The relevance is considered low if it is equal to the minimum λ_{min}:

$$\lambda_{min} = \frac{b}{M + N + a + 1} \tag{17}$$

After the learning process, only the columns of $\boldsymbol{W}$ with a high relevance (high λ_k value) are retained. This updating scheme is pursued in the results section.

4. EXPERIMENTAL SETUP

4.1. Database Grabo

To test the proposed method, a database was created where 11 users were asked to control a service robot uttering 36 simple commands, like *'quickly drive a little bit forward'* or *'turn around slowly'*. Each user had to translate this sentence in his own mother tongue in order to induce variation in the phrasing of the commands. Each command has been recorded 15 times, each time using the same phrasing. The database of each user is then split in several train/test datasets with different training sizes. 9 different training set sizes (going from 10% to 90% of the total database) were chosen. The rest of the database is used as a testing dataset. For each size 6 different subsets are randomly selected. The number of training examples for each of the 36 unique commands is the same in all versions of a particular size. For each user there are thus $9 \cdot 6 = 54$ train/test datasets, for each of these datasets an experiment is done.

It was preferred to complement the voice recordings with simulations of the action data to have better control over the distribution of the action data. The action data was randomly selected from Gaussian distributions with pre-defined means according to the different commands. The range for all the parameters is $[0; 200]$, the variance for all the distributions is chosen as 25.

4.2. Experiments

To test the performance of CSNMF it is compared to ordinary NMF. Ordinary NMF is considered to be a special case of CSNMF where the variance of the distributions in the interdependency matrix are chosen as 0, making $\boldsymbol{L}$ equal to the identity matrix.

Two performance metrics are used. First the command is classified as a command type. The percentage of correctly classified command types is the first performance metric. Next the parameter that has an ordinal structure is classified. The error of the estimated parameter value is considered to be the difference to the mean from which it is generated because this is what the robot should learn (and not the noise). The Mean Absolute Error (MAE) is the second performance metric. For each training set size the average result is calculated for all the different users and commands.

Two user behaviours are simulated. The first being a user that uses well separated clusters for the parameters. The means of the distributions where the parameters are generated from lay far apart from each other. The second behaviour is a user that uses clusters that partially overlap, i.e. lay close to each other. The distance between the means of the distributions is two times the standard deviation of the distributions. To create the interdependency matrix the variance of the user has to be estimated. To test the effect of incorrectly estimating the variance of the users, three situations are tested: the variance is correctly estimated, the variance is estimated as half the correct variance or the variance is estimated as double the correct variance.

5. RESULTS

5.1. Command type recognition

The results for both user behaviours for command type recognition are similar so only the results for the overlapping clusters are discussed here. The results are shown in figure 2. The results show that all the tests where CSNMF is used the percentage of correctly classified command types is higher than when ordinary NMF is used. The recognition accuracy for ordinary NMF varies a lot and doesn't really improve with a larger training set size. When CSNMF is used the results do ameliorate with an increasing training set size. This can be explained by the fact that CSNMF groups the parameter

classes that are close to each other together and thus has more examples per command than the ordinary NMF which interprets small variations in the demonstrations as completely different commands. The results also show that incorrectly estimating the variance does not have a huge effect on the number of correctly classified command types.

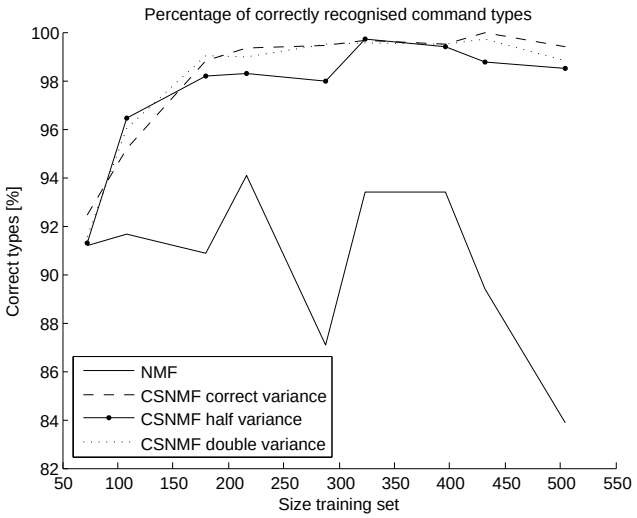

Fig. 2. Percentage of correctly classified command types for well separated clusters

5.2. MAE

The results for the MAE for both user behaviours can be seen in figures 3 and 4. The MAE for ordinary NMF does improve with an increasing number of examples but slower than the MAE for CSNMF. In general the MAE for CSNMF is considerably lower than the MAE for ordinary NMF. When the clusters overlap the difference in performance is smaller than when the clusters are separated. This is because a parameter class in ordinary NMF is not influenced by other classes close by in the ordinal structure. Ordinary NMF will only be affected by overlapping clusters if exactly the same parameter class is selected by two different clusters. Classes in CSNMF are influenced by other classes close by in the ordinal structure. CSNMF will thus be affected when two different clusters select parameter values close to each other. The negative effect is thus more severe for CSNMF. Notice that the MAE is a lot lower for overlapping clusters. This is because the clusters are closer to each other. A smaller range of parameter values is used causing the MAE to be lower.

When using well separated clusters the MAE is lower when the variance is overestimated. Indeed, even if the variance is overestimated there is no influence from other clusters. However, when using overlapping clusters overestimating does not perform better any more because now there is more influence from other clusters. This has a negative effect on the MAE. In general neither overestimating nor underestimating the variance causes CSNMF to perform much worse.

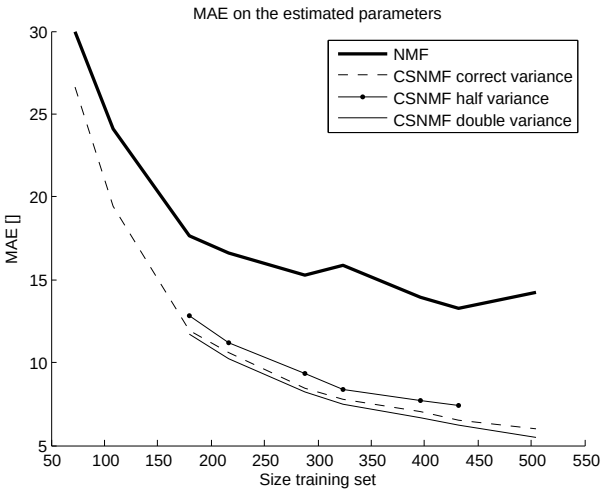

Fig. 3. MAE on the the estimation of the parameters for well separated clusters

5.3. Model order

The original model order and the resulting model order are shown in figure 5. Only the results for well separated clusters are discussed because the results for overlapping clusters are similar. The original model order is the model order chosen by the designer before training. It is computed by counting the number of used parameter values in the training data. This causes the original model order to increase with an increasing number of examples. The resulting model order is the number of columns in $\boldsymbol{W}$ that are retained after training with ARD. The ideal case would be that the resulting model order would converge to some 'correct' model order. A 'correct' model order means that it is dependent on the amount of keywords that the user uses and not the amount of training examples. If the same wording is consistently used this is equal to the amount of data clusters used in the demonstrations.

The model order does increase significantly slower with the training set size than the original model order but it doesn't quite converge either. The automatic model order selection does work well but is not yet ideal. Improving the automatic model order selection is thus a topic for further research.

The resulting model order for CSNMF is lower than for ordinary NMF. With its high model order, ordinary NMF learns a word representation for each of the 201 categories and therewith learns the acoustic representation of the same phrase (e.g. 'a little bit') multiple times without linking them semantically. In CSNMF, the model order is lower because the categories are softly grouped together under a single command. If a higher variance is chosen for the distributions in the interdependency matrix, the model order is lower because more parameters values are grouped together.

6. CONCLUSION

In this paper a learning scheme for ordinal data in a robot speech acquisition system has been proposed. Weakly supervised Non-

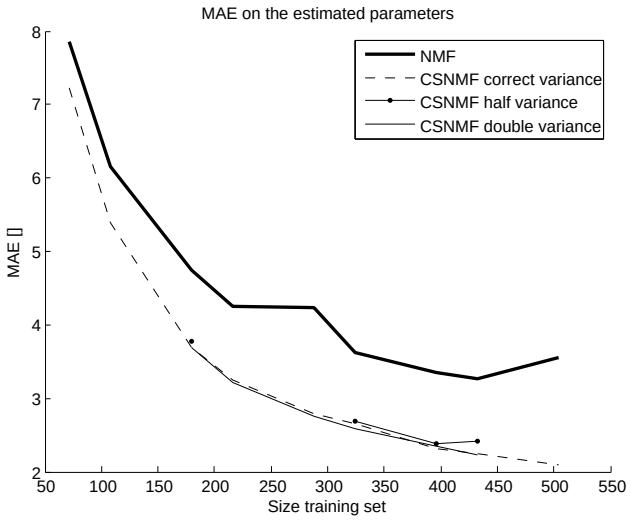

Fig. 4. MAE on the the estimation of the parameters for overlapping clusters

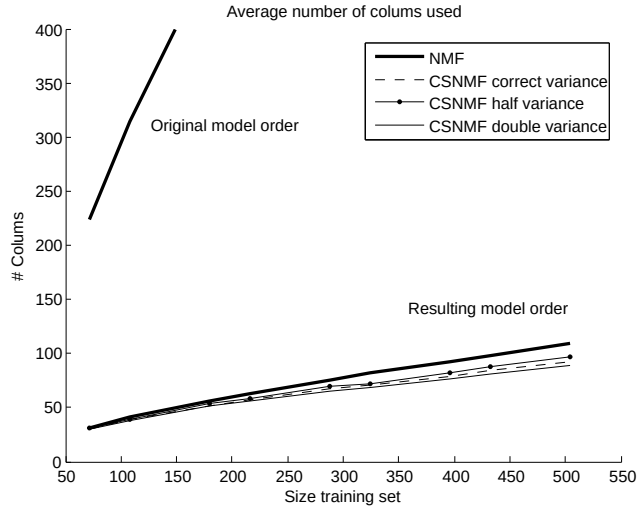

Fig. 5. Original model order and resulting model order for well separated clusters

negative Matrix Factorization is extended to take the ordinal structure of the data into account. A probabilistic treatment of the problem is presented resulting in Constrained Subspace NMF. The proposed method has been tested for two user behaviours and three variance estimations. For both user behaviours CSNMF performed better than ordinary NMF and showed to be robust against incorrect variance estimates.

7. REFERENCES

[1] Tatsuya Kawahara, Chin-Hui Lee, and Biing-Hwang Juang, "Flexible speech understanding based on combined key-phrase detection and verification," *Speech and Audio Processing, IEEE Transactions on*, vol. 6, no. 6, pp. 558–568, 1998.

[2] Jort F Gemmeke, Janneke Van De Loo, Guy De Pauw, Joris Driesen, Hugo Van hamme, and Walter Daelemans, "A self-learning assistive vocal interface based on vocabulary learning and grammar induction," in *INTERSPEECH*, 2012.

[3] Daniel D Lee and H Sebastian Seung, "Algorithms for non-negative matrix factorization," in *Advances in neural information processing systems*, 2001, pp. 556–562.

[4] Jort Gemmeke, Siddharth Sehgal, and Stuart Cunningham, "Fast vocabulary learning for disordered speech vocal interfaces," in *Spoken Language Technology Workshop (SLT)*. IEEE, 2014.

[5] Hugo Van hamme, "Hac-models: a novel approach to continuous speech recognition," in *Proceedings Interspeech, ISCA*. Citeseer, 2008.

[6] Vincent YF Tan and Cédric Févotte, "Automatic relevance determination in nonnegative matrix factorization with the β-divergence," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 7, pp. 1592–1605, 2013.

[7] Wei Chu and Zoubin Ghahramani, "Gaussian processes for ordinal regression," in *Journal of Machine Learning Research*, 2005, pp. 1019–1041.

[8] Shirish Krishnaj Shevade and Wei Chu, "Minimum enclosing spheres formulations for support vector ordinal regression," in *Data Mining, 2006. ICDM'06. Sixth International Conference on*. IEEE, 2006, pp. 1054–1058.

[9] Jianlin Cheng, Zheng Wang, and Gianluca Pollastri, "A neural network approach to ordinal regression," in *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*. IEEE, 2008, pp. 1279–1284.

[10] Rich Caruana, Shumeet Baluja, and Tom Mitchell, "Using the future to" sort out" the present: Rankprop and multitask learning for medical risk evaluation," *Advances in neural information processing systems*, pp. 959–965, 1996.

[11] Tobias Bocklet, Andreas Maier, Josef G Bauer, Felix Burkhardt, and Elmar Noth, "Age and gender recognition for telephone applications based on gmm supervectors and support vector machines," in *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*. IEEE, 2008, pp. 1605–1608.

[12] Ieva Zeltmate and Janis Grundspenkis, "An extension of frame-based knowledge representation schema," in *Proc. of the Int. Multi-Conf. on Complexity, Informatics and Cybernetics*, vol. 1, pp. 6–9.

[13] Cédric Févotte and A Taylan Cemgil, "Nonnegative matrix factorizations as probabilistic inference in composite models," in *Proc. EUSIPCO*. Citeseer, 2009, vol. 47, pp. 1913–1917.

[14] Alberto Pascual-Montano, Jose Maria Carazo, Kieko Kochi, Dietrich Lehmann, and Roberto D Pascual-Marqui, "Non-smooth nonnegative matrix factorization (nsnmf)," *Pattern*

Analysis and Machine Intelligence, IEEE Transactions on, vol. 28, no. 3, pp. 403–415, 2006.

[15] Andrzej Cichocki, Rafal Zdunek, Anh Huy Phan, and Shun-ichi Amari, *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*, John Wiley & Sons, 2009.